

A Novel Approach for Solving an Arbitrary Sparse Linear System

Minwoo Chae¹ and Stephen G. Walker¹

¹ *Department of Mathematics, University of Texas at Austin*

September 5, 2016

Abstract

It has been an open problem [Saad (2003)] to find an iterative method that can solve an arbitrary sparse linear system $Ax = b$ in an efficient way (*i.e.* guaranteed convergence at geometric rate). We propose a novel iterative algorithm which can be applied to a large sparse linear system and guarantees convergence for any consistent (*i.e.* it has a solution) linear system. Moreover, the algorithm is highly stable, fast, easy to code and does not require further constraints on A other than that a solution exists. We compare with the Krylov subspace methods which do require further constraints, such as symmetry or positive definiteness.

Keywords: Indefinite matrix, iterative method, Kullback-Leibler divergence, sparse linear system.

1 Introduction

For a given $m \times m$ matrix $A = (a_{ij})$ and a vector $b \in \mathbb{R}^m$, consider the system of linear equations

$$Ax = b. \tag{1}$$

If A is nonsingular with inverse matrix A^{-1} , there exists a unique solution to (1), denoted by $x^* = A^{-1}b$. When the dimension m is large, however, finding the inverse matrix A^{-1} is computationally unfeasible. As alternatives, a number of iterative methods have been proposed to find a sequence (x_n) approximating x^* , and they are often implementable when A is sparse, that is, most a_{ij} 's are zero. For reviews of these iterative methods within a unified framework, we refer to the monograph [18]. We start with a brief introduction of the most widely used iterative methods for solving (1).

Current iterative methods are coordinate-wise updating algorithms. Two of the most well-known methods are *Jacobi* and *Gauss-Seidel* which can be found in most standard textbooks. Given $x_n = (x_{n,j})$, the Jacobi and Gauss-Seidel methods, update x_{n+1} via

$$x_{n+1,j} = \frac{1}{a_{jj}} \left(b_j - \sum_{i \neq j} a_{ji} x_{n,i} \right)$$

and

$$x_{n+1,j} = \frac{1}{a_{jj}} \left(b_j - \sum_{i=1}^{j-1} a_{ji}x_{n+1,i} - \sum_{i=j+1}^m a_{ji}x_{n,i} \right),$$

respectively. Although they are simple and convenient, both of them are restrictive in practice because x_n is not generally guaranteed to converge x^* ; see [18].

The *Krylov subspace methods*, which are based on the *Krylov subspace* of \mathbb{R}^m

$$\mathcal{K}_n = \text{span}\{r_0, Ar_0, A^2r_0, \dots, A^{n-1}r_0\},$$

are the dominant approaches, where x_0 is an initial guess and $r_0 = b - Ax_0$. Under the assumption that A is sparse, matrix-vector multiplication is cheap to compute, so it is not difficult to handle \mathcal{K}_n even when m is very large. If A is symmetric and positive definite (SPD), the standard choice for solving (1) is the *conjugate gradient method* (CG; [12]). This is an orthogonal projection method ([18]) onto \mathcal{K}_n , finding $x_n \in x_0 + \mathcal{K}_n$ such that $b - Ax_n \perp \mathcal{K}_n$. To be more specific, recall that two vectors $u, v \in \mathbb{R}^m$ are called *A-conjugate* if $u^T Av = 0$. If A is symmetric and positive definite, then this quadratic form defines an inner product, and there is a basis for \mathbb{R}^m consisting of mutually *A-conjugate* vectors. The CG method sequentially generates mutually *A-conjugate* vectors p_1, p_2, \dots , and approximates $x^* = \sum_{j=1}^m \alpha_j p_j$ as $x_n = \sum_{j=1}^n \alpha_j p_j$, where $\alpha_j = p_j^T b / p_j^T A p_j$. Using the symmetry of A , the computation can be simplified as in Algorithm 2. Here $\|\cdot\|_q$ denotes the ℓ_q -norm on \mathbb{R}^m .

Algorithm 1 Conjugate gradient method for SPD A

- 1: Initialize x_0 ; set $j \leftarrow 0$, $r_0 \leftarrow b - Ax_0$, $p_0 \leftarrow r_0$ and a constant $\epsilon_{\text{tol}} > 0$
 - 2: **while** $\|r_j\|_2 > \epsilon_{\text{tol}}$ **do**
 - 3: $\alpha_j \leftarrow r_j^T r_j / p_j^T A p_j$
 - 4: $x_{j+1} \leftarrow x_j + \alpha_j p_j$
 - 5: $r_{j+1} \leftarrow r_j - \alpha_j A p_j$
 - 6: $\beta_j \leftarrow r_{j+1}^T r_{j+1} / r_j^T r_j$
 - 7: $p_{j+1} \leftarrow r_{j+1} + \beta_j p_j$
 - 8: $j \leftarrow j + 1$
 - 9: **return** x_j
-

For a general matrix A , the generalized minimal residual method (GMRES; [20]) is the most popular. It is an oblique projection method ([18]), which finds $x_k \in \mathcal{K}_k$ satisfying $b - Ax_k \perp A\mathcal{K}_k$, where $A\mathcal{K}_k = \{Av : v \in \mathcal{K}_k\}$. When implementing GMRES, Arnoldi's method ([1]) is applied for computing an orthonormal basis of \mathcal{K}_k . The method can be written as in Algorithm 2. For a given initial x_0 , let us write the result of Algorithm 2 as $\text{GMRES}_k(x_0)$. Since the computational cost of Algorithm 2 is prohibitive for large k , a restart version of GMRES_k , defined as $x_{n+1} = \text{GMRES}_k(x_n)$, is applied with small k .

We denote this restart version as GMRES(k). It should be noted that the generalized conjugate residual (GCR; [7]), ORTHODIR ([22]) and Axelsson's method ([2]) are mathematically equivalent to GMRES; but it is known ([20]) that GMRES is computationally more efficient and reliable. More connections between these methods are discussed in [19]. Convergence is guaranteed but there are restrictions; see Section 3.

Algorithm 2 GMRES _{k}

```

1: Initialize  $x_0$ ; set  $\beta \leftarrow \|b - Ax_0\|_2$ ,  $v_1 \leftarrow (b - Ax_0)/\beta$  and a constant  $\epsilon_{\text{tol}} > 0$ 
2: for  $j = 1, \dots, k$  do
3:    $w_j \leftarrow Av_j$ 
4:   for  $i = 1, \dots, j$  do
5:      $h_{ij} \leftarrow w_j^T v_i$ 
6:      $w_j \leftarrow w_j - h_{ij}v_i$ 
7:    $h_{j+1,j} \leftarrow \|w_j\|_2$ 
8:   if  $h_{j+1,j} < \epsilon_{\text{tol}}$  then set  $k \leftarrow j$  and break
9:    $v_{j+1} \leftarrow w_j/h_{j+1,j}$ 
10:  $y_k \leftarrow \operatorname{argmin}_y \| \beta e_1 - H_k y \|_2$ , where  $H_k = (h_{ij})_{i \leq k+1, j \leq k}$  and  $e_1 = (1, 0, \dots, 0)^T$ 
11:  $x_k = x_0 + V_k y_k$ , where  $V_k = (v_1, \dots, v_k) \in \mathbb{R}^{m \times k}$ 
12: return  $x_k$ 

```

The minimum residual method (MINRES; [16]) can be understood as a special case of GMRES when A is a symmetric matrix. In this case, Arnoldi's method (steps 2-9) in Algorithm 2 can be replaced by the simpler Lanczos algorithm ([14]), described in Algorithm 3, where $\alpha_j = h_{jj}$ and $\beta_j = h_{j-1,j}$.

Algorithm 3 Lanczos algorithm

```

1: Set  $\beta_1 \leftarrow 0$  and  $v_0 \leftarrow 0$ 
2: for  $j = 1, \dots, k$  do
3:    $w_j \leftarrow Av_j - \beta_j v_{j-1}$ 
4:    $\alpha_j \leftarrow w_j^T v_j$ 
5:    $w_j \leftarrow w_j - \alpha_j v_j$ 
6:    $\beta_{j+1} \leftarrow \|w_j\|_2$ 
7:   if  $\beta_{j+1} < \epsilon_{\text{tol}}$  then set  $k \leftarrow j$  and break
8:    $v_{j+1} \leftarrow w_j/\beta_{j+1}$ 

```

In summary, standard approaches for solving (1) are (i) CG for SPD A ; (ii) MINRES for symmetric A ; and (iii) GMRES for general A , however convergence is not guaranteed for GMRES. There is a large amount of other general approaches, and many of them are variations and extensions of Krylov subspace methods. Each method has some appealing properties, but it is difficult in general to analyze them theoretically. See Chapter 7 of [18]. Also, there are some algorithms which are devised to solve a structured linear system

[3, 9, 13]. To the best of our knowledge, however, there is no efficient iterative algorithm that can solve an arbitrary sparse linear system. In particular, the most popular, GMRES, often has quite strange convergence properties [8, 11] making the algorithm difficult to use in practice. As a consequence, preconditioning is an important issue, and there is no precise rule governing it.

In this paper, we propose an iterative method which guarantees convergence with a geometric rate for any nonsingular A . The algorithm is highly stable, fast, easy to implement and requires small storage. The proposed algorithm can be a good alternative to the Krylov subspace methods. The new algorithm and its theoretical properties are studied in Section 2. Comparison of computational complexity and known convergence properties are provided in Section 3. Concluding remarks are given in Section 4.

Before proceeding, it is useful to set some notation. Every vector such as b and x are column vectors, and components are denoted with a subscript, *i.e.* $b = (b_i)$. Dots in subscripts present the summation in those indices, *i.e.* $a_{.j} = \sum_{i=1}^m a_{ij}$. The i th row and j th column of A are denoted by \mathbf{r}_i and \mathbf{c}_j , respectively. For X , which may be a vector or a matrix, is said to be nonnegative (positive, resp.) and denoted $X \geq 0$ ($x > 0$, resp.) if each component of X is nonnegative (positive, resp.). For $X \in \mathbb{R}^{m \times m}$, $X \succeq 0$ ($X \succ 0$, resp.) represents that X is nonnegative (positive, resp.) definite, *i.e.* $u^T X u \geq 0$ ($u^T X u > 0$, resp.) for every nonzero vector u . The number of nonzero elements of X is denoted \mathcal{N}_X .

2 An iterative algorithm with guaranteed convergence

In the first subsection, we develop an iterative algorithm that can solve an arbitrary nonnegative linear system. This is based on a proposal of Walker [21] for nonnegative A, b and x^* . For a general linear system, we prove in the second subsection that the system (1) can be embedded into a larger system $Py = c$, where $P \geq 0$ and $\mathcal{N}_P = \mathcal{N}_A + 2J$, where J is an integer less than or equal to m . Then, the iterative algorithm designed for a nonnegative linear system can be applied to solve the larger system. Illustrative examples are presented in the last subsection.

2.1 Algorithm for solving nonnegative systems

Assume that A, b and x^* are nonnegative. In this case, Walker [21] proposed an iterative algorithm given by

$$x_{n+1,j} = \frac{x_{n,j}}{a_{.j}} \sum_{i=1}^m a_{ij} \frac{b_i}{b_{n,i}}, \quad n \geq 0, \quad (2)$$

where $b_n = (b_{n,i}) = Ax_n$ and $x_0 \geq 0$ is an initial guess. We say that $v \in \mathbb{R}^m$ is a probability vector if $v_i \geq 0$ and $\sum_{i=1}^m v_i = 1$. Assume that b, x and \mathbf{c}_j , $1 \leq j \leq m$, are probability vectors, and consider the discrete random variables I and J whose joint distribution is given by

$$\mathbb{P}(J = j) = x_j, \quad \mathbb{P}(I = i | J = j) = a_{ij} \quad \text{for } 1 \leq i, j \leq m.$$

The marginal distribution of I is given as

$$\mathbb{P}(I = i) = \sum_{j=1}^m \mathbb{P}(I = i|J = j)\mathbb{P}(J = j) = \sum_{j=1}^m x_j a_{ij} = b_i.$$

By Bayes theorem

$$\mathbb{P}(J = j|I = i) = \frac{\mathbb{P}(J = j)\mathbb{P}(I = i|J = j)}{\sum_{j'=1}^m \mathbb{P}(J = j')\mathbb{P}(I = i|J = j')} = \frac{x_j a_{ij}}{\mathbf{r}_i^T \mathbf{x}},$$

so it follows that

$$x_j = \mathbb{P}(J = j) = \sum_{i=1}^m \mathbb{P}(J = j|I = i)\mathbb{P}(I = i) = x_j \sum_{i=1}^m \frac{a_{ij} b_i}{\mathbf{r}_i^T \mathbf{x}}.$$

For a given current guess x_n of x^* , this leads to the update (2).

If $A, b, x \geq 0$ but some of b, x and \mathbf{c}_j 's are not probability vectors, we can easily reformulate the problem as

$$\tilde{A}\tilde{x} = \tilde{b} \tag{3}$$

with the update (2), where $\tilde{A} = (a_{ij}/a_{.j})_{i,j \leq m}$, $\tilde{x} = (x_j a_{.j}/b.)_{j=1}^m$ and $\tilde{b} = (b_i/b.)_{i=1}^m$.

If A is nonsingular, Theorem 2.1 assures the convergence of the update (2) with geometric rate. We need well-known bounds for probability metrics for the proof. For m -dimensional vectors $u, v \geq 0$, define the *Kullback-Leibler (KL) divergence* $D(u, v) = \sum_{i=1}^m u_i \log(u_i/v_i)$ and *total variation* $V(u, v) = \sum_{i=1}^m |u_i - v_i|$. In the definition of the KL divergence, we let $u_i \log(u_i/v_i) = 0$ if $u_i = 0$ and $D(u, v) = \infty$ if $u_i > 0$ and $v_i = 0$ for some i . It is well-known that $D(u, v) \geq 0$ for every pair of probability vectors (u, v) , and equality holds if and only if $u = v$.

Theorem 2.1. *Assume that $A \geq 0$, $x^*, b > 0$ and A is nonsingular. If $x_0 > 0$, then x_n defined by (2) satisfies $D(\tilde{x}^*, \tilde{x}_n) \leq C(1 - \delta)^n$ for some constants $C, \delta > 0$, where $\tilde{x}^* = (x_j^* a_{.j}/b.)_{j=1}^m$ and $\tilde{x}_n = (x_{n,j} a_{.j}/b.)_{j=1}^m$.*

Proof. If some of b, x^* and \mathbf{c}_j 's are not probability vectors, we can reformulate the problem using (3). Therefore, we may assume without loss of generality that b, x^* and \mathbf{c}_j 's are probability vectors. For any $x_0 > 0$, it is easy to see that $x_n > 0$ and $\sum_{j=1}^m x_{n,j} = 1$ for every $n \geq 1$. Thus, b_n and x_n are also probability vectors for every $n \geq 1$. From (2) we have

$$\log x_{n+1,j} = \log x_{n,j} + \log \sum_{i=1}^m \left(\frac{b_i}{b_{n,i}} a_{ij} \right) \geq \log x_{n,j} + \sum_{i=1}^m a_{ij} \log \left(\frac{b_i}{b_{n,i}} \right),$$

where the inequality holds by Jensen. Therefore,

$$\sum_{j=1}^m x_j^* \log x_{n+1,j} \geq \sum_{j=1}^m x_j^* \log x_{n,j} + D(b, b_n).$$

This implies that

$$D(x^*, x_{n+1}) \leq D(x^*, x_n) - D(b, b_n), \quad (4)$$

and $D(x^*, x_n)$ converges, by the monotone convergence theorem. Thus, $D(b, b_n) \rightarrow 0$, which in turn implies that $x_n \rightarrow x^*$.

Note that

$$V(x^*, x_n) = V(A^{-1}b, A^{-1}b_n) \leq \|A^{-1}\|_1 V(b, b_n),$$

where $\|\cdot\|_1$ denotes the ℓ_1 -operator norm (*i.e.* maximum absolute column sum) of the matrix. Therefore,

$$D(b, b_n) \geq \frac{1}{2} V^2(b, b_n) \geq \frac{1}{2\|A^{-1}\|_1^2} V^2(x^*, x_n),$$

where the first inequality holds by Pinsker [5, 17]. Since

$$\begin{aligned} D(x^*, x_n) &= \sum_{j=1}^m x_j^* \log \frac{x_j^*}{x_{n,j}} \leq \sum_{j=1}^m x_j^* \left(\frac{x_j^*}{x_{n,j}} - 1 \right) = \sum_{j=1}^m \left(1 + \frac{x_j^* - x_{n,j}}{x_{n,j}} \right) (x_j^* - x_{n,j}) \\ &= \sum_{j=1}^m \frac{(x_j^* - x_{n,j})^2}{x_{n,j}} \leq \left(\sum_{j=1}^m \frac{|x_j^* - x_{n,j}|}{\sqrt{x_{n,j}}} \right)^2 \leq V^2(x^*, x_n) \max_{1 \leq j \leq m} x_{n,j}^{-1} \end{aligned}$$

and $x_n \rightarrow x^*$, we have $D(b, b_n) \geq \delta D(x^*, x_n)$ for all large enough n , where

$$\delta = \frac{1}{3\|A^{-1}\|_1^2} \min_{1 \leq j \leq m} x_{n,j}^*.$$

Therefore, by (4),

$$\delta D(x^*, x_n) \leq D(b, b_n) \leq D(x^*, x_n) - D(x^*, x_{n+1})$$

for all large enough n . It follows that $D(x^*, x_{n+1}) \leq (1 - \delta)D(x^*, x_n)$ for large all enough n , and this completes the proof. \square

The key to the proof of Theorem 2.1 is inequality (4). This inequality implies that the larger $D(b, b_n)$ is the larger we gain at the n th iteration. Thus, the most important factor determining the convergence rate of $D(x^*, x_n)$ is $\|A^{-1}\|_1$. On the other hand, by inequality (4), slower convergence of $D(x^*, x_n)$ implies that $D(b, b_n)$ is already small, which is also a good convergence criterion.

It should be noted that $x^*, b > 0$ is essential for the convergence of the algorithm. When $b_i \leq 0$ for some i , we can easily reformulate the problem as

$$Ax_t = b_t, \quad (5)$$

where $x_t = x + t\mathbf{1}_m$, $b_t = b + tA\mathbf{1}_m$, $\mathbf{1}_m = (1, \dots, 1)^T$ and $t > 0$ is a constant such that $b_t > 0$. Note also that $A \geq 0$ and $b > 0$ does not imply that $x \geq 0$. If t is large enough, however, we have $x^* + t\mathbf{1}_m > 0$, leading to Algorithm 4 which guarantees the convergence for any $A \geq 0$ and $b > 0$. We call this algorithm as the *nonnegative algorithm (NNA)*.

Algorithm 4 Nonnegative algorithm for $A \geq 0$ and $b > 0$

```
1: Initialize  $x > 0$ ; set  $k = 0$  and constants  $t, \epsilon_{\text{tol}} > 0$ 
2: while convergence criterion not satisfied do
3:   Update  $x$  as (2)
4: if  $\|Ax - b\|_2 > \epsilon_{\text{tol}}$  then
5:    $b \leftarrow b + tA\mathbf{1}_m$ 
6:    $x \leftarrow x + t\mathbf{1}_m$ 
7:    $k \leftarrow k + 1$ 
8:   goto line 2
9: return  $x - kt\mathbf{1}_m$ 
```

Here we consider the computational complexity of Algorithm 4. In (2), we first need to compute $b_n = Ax_n$, and then compute $c = b/b_n$, where $/$ represents componentwise division. Finally, we compute $x_{n+1} = (\tilde{A}^T c) \circ x_n$, where \circ denotes componentwise multiplication and $\tilde{A} = (a_{ij}/a_{.j})_{i,j \leq m}$. In summary, we need two matrix-vector multiplications and two vector-vector componentwise operations. Note that the number of flops (floating-point operations; addition, subtraction, multiplication, or division) for matrix multiplication is less than $2\mathcal{N}_A$. Also, for a vector-vector multiplication (or division), $2m$ flops are required. Therefore, the total number of flops for one iteration of (2) is less than $4(\mathcal{N}_A + m)$. We compare the number of flops with other algorithms in Section 3.

We can apply Algorithm 4 for any consistent linear system even when A is not invertible. For the remainder of this subsection, we assume that $A \in \mathbb{R}^{m_1 \times m_2}$, $b \in \mathbb{R}^{m_2}$, $x \in \mathbb{R}^{m_1}$ and x^* is a (not necessarily unique) solution of the linear system $Ax = b$. Theorem 2.2 assures the convergence for any consistent linear system. Furthermore, it implies that the number N of iterations for achieving $D(b_N, b) \leq \epsilon$ is at most $O(1/\epsilon)$.

Theorem 2.2. Assume that $A \geq 0$, $x^*, b > 0$ and $Ax^* = b$. For any $x_0 > 0$, the sequence (x_n) defined as

$$x_{n+1,j} = \frac{x_{n,j}}{a_{.j}} \sum_{i=1}^{m_1} a_{ij} \frac{b_i}{b_{n,i}}, \quad n \geq 0, 1 \leq j \leq m_2$$

satisfies $b_n \rightarrow b$, where $b_n = Ax_n$. In particular, for every $\epsilon > 0$ there exists $N \leq D(x^*, x_1)/\epsilon + 1$ such that $D(b_N, b) \leq \epsilon$.

Proof. As in the proof of Theorem 2.1, we may assume that x^*, b and \mathbf{c}_j , $1 \leq j \leq m_2$ are probability vectors without loss of generality. Then, b_n and x_n are probability vectors for every $n \geq 1$, so the inequality (4) holds in the same way. Thus, $D(x^*, x_n)$ converges by the monotone convergence theorem. It follows that $D(b, b_n) \rightarrow 0$.

For given $\epsilon > 0$, let N be the largest integer less than or equal to $D(x^*, x_1)/\epsilon + 1$. Assume that $D(b, b_n) > \epsilon$ for every $n \leq N$. Then, since

$$0 \leq D(x^*, x_{N+1}) \leq D(x^*, x_1) - \sum_{n=1}^N D(b, b_n)$$

by (4), we have $N < D(x^*, x_1)/\epsilon$. This makes a contradiction and completes the proof. \square

2.2 General linear systems

For convenience, we only consider a square matrix A , but the approach introduced in this subsection can also be applied to any consistent linear system. The main idea is to embed the original system (1) into a larger nonnegative system, and then apply Algorithm 4. The enlarged system should be minimal to reduce any additional computational burden.

As an illustrative example, consider the system of linear equations

$$\begin{aligned} a_{11}x_1 - a_{12}x_2 + a_{13}x_3 &= b_1, \\ a_{21}x_1 + a_{22}x_2 - a_{23}x_3 &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3, \end{aligned}$$

where $a_{ij} \geq 0$ for every i and j , so A has negative elements. We consider two more equations

$$x_2 + x_4 = 0 \quad \text{and} \quad x_3 + x_5 = 0,$$

where each equation contains only two nonzero elements. Then, it is easy to see that solving the linear system consisting of the above five equations is equivalent to solving the following five equations:

$$\begin{aligned} a_{11}x_1 &+ a_{13}x_3 + a_{12}x_4 &= b_1, \\ a_{21}x_1 + a_{22}x_2 &+ a_{23}x_5 &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & &= b_3, \\ x_2 &+ x_4 &= 0, \\ x_3 &+ x_5 &= 0. \end{aligned} \tag{6}$$

Let $Py = c$ be the matrix form of (6), then we have $P \geq 0$, so NNA can be applied. This can be generalized as in the following theorem.

Theorem 2.3. *For $A \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$, assume that $Ax^* = b$. Then, there exists a linear system $Py = c$ with solution y^* , such that P is a $(m + J) \times (m + J)$ matrix with $J \leq m$, $\mathcal{N}_P = \mathcal{N}_A + 2J$ and the first m components of y^* are equal to x^* .*

Proof. Let $\mathcal{J} = \{j \leq m : a_{ij} < 0 \text{ for some } i \leq m\}$ and J be the cardinality of \mathcal{J} . If $J > 0$, we can write $\mathcal{J} = \{j_1, \dots, j_J\}$ with $j_1 < \dots < j_J$. Let $A^+ = (\max\{a_{ij}, 0\})_{i,j \leq m}$, $A^- = -(\min\{a_{ij}, 0\})_{i,j \leq m}$ and \tilde{A}^- be the $m \times J$ sub-matrix of A^- consisting of all nonzero columns. Let $D = (d_{ij})$ be the $J \times m$ matrix defined as

$$d_{ij} = \begin{cases} 1 & \text{if } j = j_i \\ 0 & \text{otherwise.} \end{cases}$$

Define a $(m + J) \times (m + J)$ matrix P as

$$P = \begin{pmatrix} A^+ & \tilde{A}^- \\ D & I_J \end{pmatrix},$$

where I_J denotes the $J \times J$ identity matrices. It is obvious that $\mathcal{N}_P = \mathcal{N}_A + 2J$. Consider the linear system

$$Py = c, \quad (7)$$

where $c = (b^T, \mathbf{0}_J^T)^T$ and $\mathbf{0}_J \in \mathbb{R}^J$ is the zero vector. Then it is easy to see that $y^* = ((x^*)^T, -(x_{\mathcal{J}}^*)^T)^T$ is a solution of (7), where $x_{\mathcal{J}}^* = (x_j^*)_{j \in \mathcal{J}}$. \square

Hence, from the proof, we see that both P and c are easy to find.

2.3 Illustrations

Given Theorem 2.3 and the use of t , without loss of generality, we assume $Ax = b$ is a nonnegative linear system, *i.e.* $A, b \geq 0$ and $x^* > 0$. In fact, we will show that t can be set arbitrarily large with no effect on convergence.

Theorem 2.1 implies that for fast convergence, $\|A^{-1}\|_1$ should be small. Before studying this, we look at convergence for different choices of t . We set $m = 10$ and generate a matrix A by sampling a_{ij} independently as the absolute value of standard normal random variables. Hence, with probability one, A will be invertible. Each component x_j^* is generated in the same way. We then ran 10000 iterations of Algorithm 4 with $t = 10, 100$ and 1000. At each step, we obtain $\|x_n - x^*\|_2$, $\|Ax_n - Ax^*\|_2$, $D(\tilde{x}^*, \tilde{x}_n)$ and $D(\tilde{A}\tilde{x}^*, \tilde{A}\tilde{x}_n)$ which are drawn in Figure 1. For large values of t , the KL-divergence tends to be small because both \tilde{x}^* and \tilde{x}_n are close to the uniform probability vectors. On the other hand, if we compare the Euclidean and residual norms with the original scale, the results are robust to the value of t . This is a common phenomenon over all our experiments. Therefore, we can choose t sufficiently large in practice.

One more important feature in Figure 1 is that the convergence speed of $\|Ax_n - b\|_2$ is much faster than that of $\|x_n - x^*\|_2$. As noted previously in terms of the KL-divergence, the convergence speed of $\|x_n - x^*\|_2$ is slow only when $\|Ax_n - b\|_2$ is small. Large $\|x_n - x^*\|_2$ and small $\|Ax_n - b\|_2$ can happen when the smallest absolute eigenvalue of A is small, in other words a matrix norm of A^{-1} is large. To empirically demonstrate this, we let $A_s = J_m + sI_m$ and sampled b_i independently from the uniform distribution on the unit interval $[0, 1]$, where I_m is the m -dimensional identity matrix, $J_m = \mathbf{1}_m \mathbf{1}_m^T$ and $m = 10$. We ran 10000 iterations of Algorithm 4 for solving $A_s x = b$ with $t = 100$ and $s = 0.05, \dots, 0.40$. Note that $\|A_s^{-1}\|_1 = 35.9$ for $s = 0.05$ and $\|A_s^{-1}\|_1 = 4.41$ for $s = 0.40$. It is clear that the convergence speed is better for smaller $\|A_s^{-1}\|_1$.

In the next section, we compare our algorithm with those mentioned in Section 1. We do this under the conditions of guaranteed convergence, which impose a restriction on all algorithms, save our own. In particular, we will compare flops per iteration and convergence rate.

Figure 1: The effect of the value of t : $t = 10$ (left), 100 (middle) and 1000 (right). Top figures depict $\|x_n - x^*\|_2$ (black solid) and $\|Ax_n - b\|_2$ (red dashed). Bottom figures represent $D(\tilde{x}^*, \tilde{x}_n)$ (black solid) and $D(\tilde{A}\tilde{x}^*, \tilde{A}\tilde{x}_n)$ (red dashed).

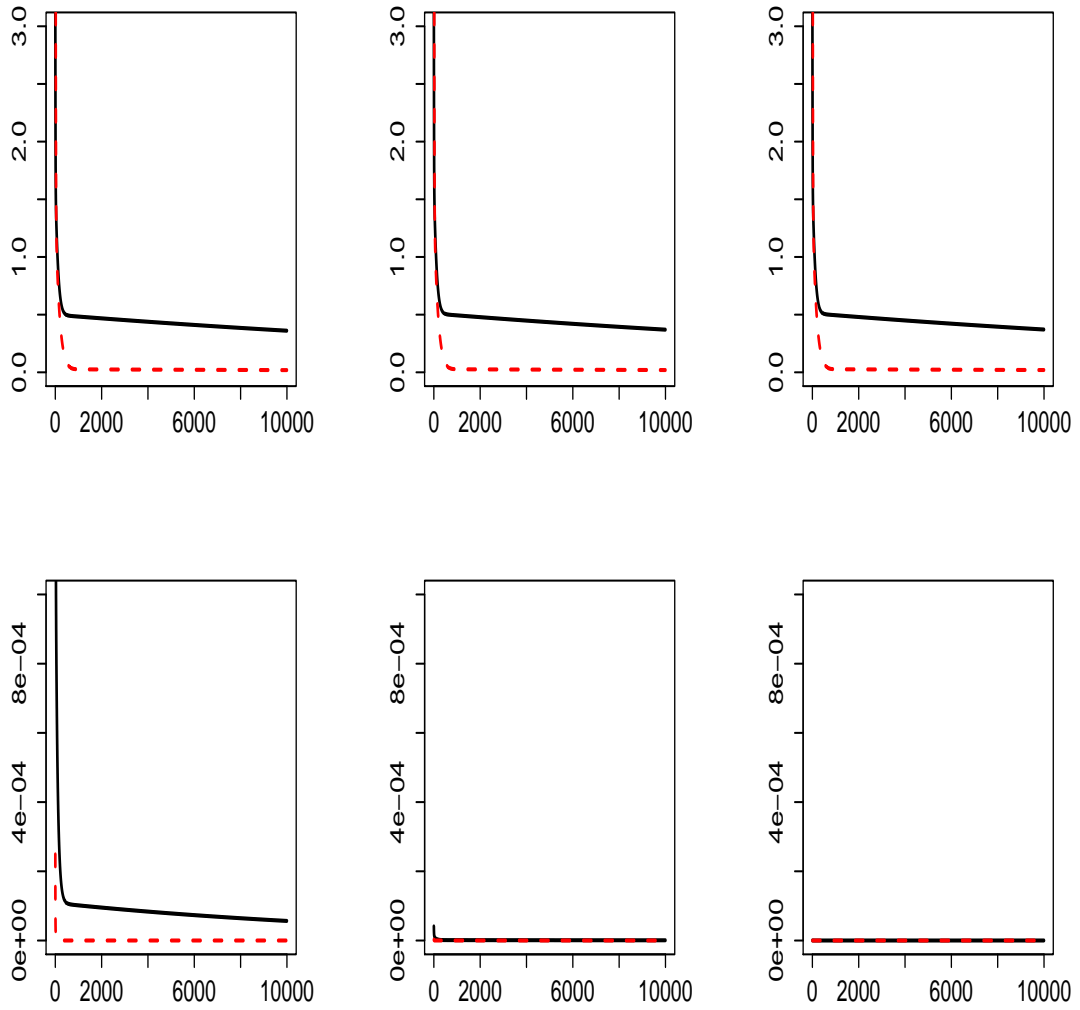
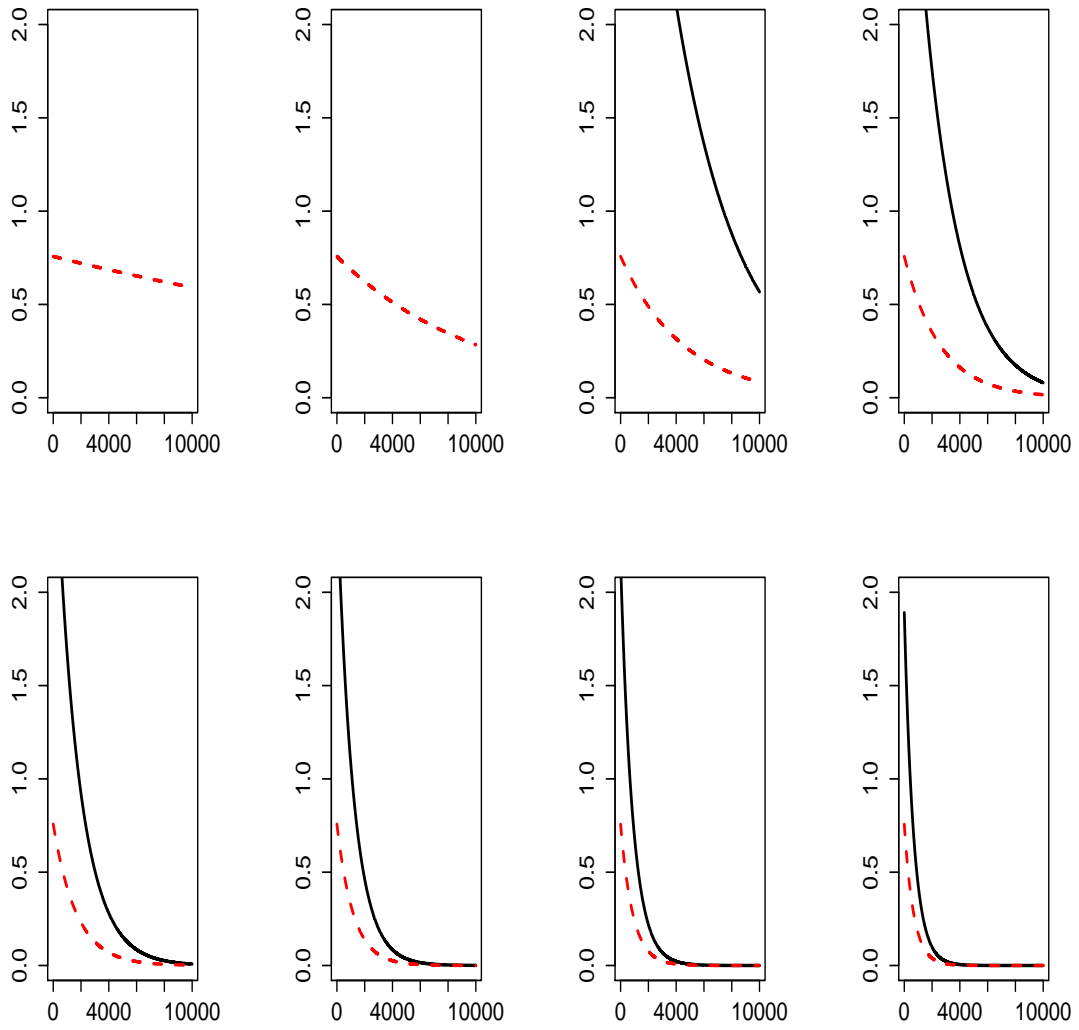


Figure 2: The effect of $\|A^{-1}\|_1$ on the convergence speed. Plots of $\|x_n - x^*\|_2$ (black solid) and $\|A_s x_n - b\|_2$ (red dashed) for $A_s = J_m + sI_m$ with $s = 0.05, \dots, 0.40$. In the first and second plots, $\|x_n - x^*\|_2$ are larger than 2.



3 Comparison with other iterative methods

As mentioned in the introduction, there is a vast amount of literature for solving sparse linear systems, but difficult to study theoretically. As a consequence, only a few algorithms possess convergence properties but even then under restrictive conditions. In this section, we compare widely used iterative methods and their convergence properties. Under the assumption that the arithmetic is exact, the result of this section is summarized in Table 1. Note that the computational complexities of MINRES(k) and GMRES(k) are not directly comparable to those of other methods because they depend on the number of step size k . The convergence rate of each method is also not directly comparable because known theoretical results are typically for the worst cases, and use different measures such as KL-divergence (NNA), Euclidean (Jacobi and Gauss-Seidel), conjugate (conjugate gradient) and residual (MINRES and GMRES) norms.

To illustrate our point, let us do a direct comparison with GMRES(k). First, the rates are not comparable. To see this, let A be a 2×2 diagonal matrix with $a_{11} = a_1$ and $a_{22} = a_2$, and assume that $a_1 \geq a_2 > 0$. In this case, $\|A^{-1}\|_1 = 1/a_2$ and

$$\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}^2((A + A^T)/2)} = \frac{a_1^2}{a_2^2},$$

so there is no rule which is the larger or smaller. However, we are better in terms of flops and storage and also have guaranteed convergence under less restrictive assumptions compared to GMRES(k).

3.1 Basic methods: Jacobi and Gauss-Seidel

It is easy to see that the numbers of flops for each step of the Jacobi and Gauss-Seidel methods are $2(N_A + m)$. Also, required storages is $N_A + 3m$ for Jacobi and $N_A + 2m$ for Gauss-Seidel. Let L, U and D be the lower, upper triangular and diagonal parts of $A = L + U + D$, respectively. Then, the Jacobi and Gauss-Seidel methods can be expressed in matrix forms as

$$x_{n+1} = D^{-1}\{b - (L + U)x_n\} \quad \text{and} \quad x_{n+1} = (L + D)^{-1}(b - Ux_n),$$

respectively. It is well-known ([18]) that updates of the form $x_{n+1} = Gx_n + f$ for some $G \in \mathbb{R}^{m \times m}$ and $f \in \mathbb{R}^m$ assures convergence if $\rho(G) < 1$, where $\rho(G)$ is the spectral radius of G . More specifically, x_n obtained by the Jacobi and Gauss-Seidel methods satisfy

$$\|x_n - x^*\|_2 \leq \{\rho(D^{-1}(L + U))\}^n \|x_0 - x^*\|_2$$

and

$$\|x_n - x^*\|_2 \leq \{\rho((L + D)^{-1}U)\}^n \|x_0 - x^*\|_2,$$

respectively. It follows that $x_n \rightarrow x^*$ if the corresponding spectral radius is strictly smaller than 1. For both methods, x_n sometimes converges to x^* even when the spectral radius is larger than 1.

Table 1: Comparison of iterative methods with known convergence properties. The first column represents sufficient conditions guaranteeing the convergence: DD (diagonally dominant), PD (positive definite) and SPD (symmetric and PD). Small constants in the third column provide good convergence rates in general.

	Conditions for convergence	Effective constants	FLOPs	Storage
NNA	-	$\ A^{-1}\ _1$	$O(N_A + m)$	$O(N_A + m)$
Jacobi	DD	$\rho(D^{-1}(L + U))$	$O(N_A + m)$	$O(N_A + m)$
Gauss-Seidel	DD or SPD	$\rho((L + D)^{-1}U)$	$O(N_A + m)$	$O(N_A + m)$
Conjugate gradient	SPD	$\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$	$O(N_A + m)$	$O(N_A + m)$
MINRES(k)	symmetric	$\frac{\lambda_{\max}(A^4)}{\lambda_{\min}^2(A^2)}$	$O(kN_A + km)$	$O(N_A + km)$
GMRES(k)	PD	$\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}^2((A + A^T)/2)}$	$O(kN_A + k^2m)$	$O(N_A + km)$

It can be expensive to compute the spectral radius of a given large matrix. Fortunately, there are well-known sufficient conditions which are easy to check. A matrix $A \in \mathbb{R}^{m \times m}$ is called *diagonally dominant* if $|a_{jj}| \geq \sum_{i \neq j} a_{ji}$ for every $i \geq 1$, and *strictly diagonally dominant* if every inequality is strict. A matrix A is called *irreducible* if the graph representation of A is irreducible, and *irreducibly diagonally dominant* if it is irreducible, diagonally dominant and $|a_{jj}| > \sum_{i \neq j} |a_{ji}|$ for some $j \geq 1$. If A is strictly or irreducibly diagonally dominant, then $\rho(D^{-1}(L + U)) < 1$ and $\rho((L + D)^{-1}U) < 1$; see [18]. Another sufficient condition for $\rho((L + D)^{-1}U) < 1$ is that A is symmetric and positive definite; see [10].

3.2 Conjugate gradient method

It is easy to see that the number of flops in steps 3–8 of Algorithm 1 is $2N_A + 12m$, and the required storage is $N_A + 4m$. Let x_n be the approximate solution obtained at the n th step of the conjugate gradient method. If the arithmetic is exact, we have $x_m = x^*$, so the exact solution can be found in m steps. If m is prohibitively large, let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ be the maximum and minimum eigenvalues of A , respectively. Then, an upper

bound on the conjugate norm between x_n and x^* is given as

$$(x_n - x^*)^T A(x_n - x^*) \leq 4 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{2n} (x_0 - x^*)^T A(x_0 - x^*),$$

where $\kappa = \lambda_{\max}(A)/\lambda_{\min}(A)$ ([18]). In practice, the improvement is typically linear in the step size; see [15].

3.3 MINRES and GMRES

Ignoring the computational complexity of step 10, that is relatively small for $k \ll m$, the numbers of flops required for steps 3, 5, 6, 7, 9 and 11 of Algorithm 2 are $2N_A, 2m, 2m, 2m, m$ and $(2k+1)m$, respectively. Thus, the number of flops is $2k\mathcal{N}_A + (2k^2 + 7k + 1)m$. Since we only need to save A , the orthonormal matrix $V_k \in \mathbb{R}^{m \times k}$, the approximate solution and vector for Av_i , the required storage is $\mathcal{N}_A + (k+2)m$. Here, storage for the Hassenberg matrix H_k is ignored because k is relatively small. For the Lanczos algorithm (Algorithm 3), it is not difficult to see that the number of flops is $k(2\mathcal{N}_A + 9m)$.

In general, Algorithm 2 does not guarantee convergence unless $k = m$. In particular, it is shown in [11] that for any decreasing sequence $\epsilon_0 > \epsilon_1 > \dots > \epsilon_m = 0$, there exists a matrix $A \in \mathbb{R}^{m \times m}$ and vectors $b, x_0 \in \mathbb{R}^m$ such that $\|\text{GMRES}_k(x_0)\|_2 = \epsilon_k$. Define (x_n) as $x_{n+1} = \text{GMRES}_k(x_n)$, a sequence generated by the restarted GMRES. Then, if $A \succ 0$, x_n converges for any $k \geq 1$; see [6]. In particular, the rate is given by

$$\|Ax_n - b\|_2^2 \leq \left\{ 1 - \frac{\lambda_{\min}^2((A + A^T)/2)}{\lambda_{\max}(A^T A)} \right\}^{nk} \|Ax_0 - b\|_2^2.$$

Some other convergence criteria of GMRES_k can be found in [4]. Also, more general upper bounds for residual norms, but not guaranteeing convergence, can be found in [15, 18].

If A is symmetric (not necessarily positive definite),

$$\|Ax_n - b\|_2^2 \leq \left\{ 1 - \frac{\lambda_{\min}^2(A^2)}{\lambda_{\max}(A^4)} \right\}^n \|Ax_0 - b\|_2^2$$

for every $k \geq 2$; see [4], assuring the convergence of restarted MINRES. Under a certain condition on the spectrum of A , a different type of upper bound can be found in [15].

3.4 s -step methods

A number of s -step methods and their convergence properties are studied in [4]. In particular, it is shown that s -step generalized conjugate residual, Orthomin(k) and minimal residual methods converge for all positive definite and some indefinite matrices. Here, s -step minimal residual method is mathematically equivalent to $\text{GMRES}(s)$. However, it is not easy in practice to check conditions for convergence of indefinite matrices. Furthermore, computational costs for s -step methods can be expensive because they require more matrix-vector multiplications in each step.

4 Discussion

The main contribution of the paper is to describe an algorithm with guaranteed convergence for indefinite linear systems of equations. We believe there is no other alternative which has the same property. Moreover, the algorithm is easy to code and implement, is stable and fast. The key idea is that nonnegative systems have unique settings which allow the existence of the algorithm and we have shown in this paper how any system can be embedded within a nonnegative system. Other algorithms, such as CG and GMRES(k), guarantee convergence under different conditions, but it is not possible in general to transform an arbitrary system into a guaranteed convergent one.

References

- [1] Arnoldi, W. E. (1951). The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29.
- [2] Axelsson, O. (1980). Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. *Linear Algebra and Its Applications*, 29:1–16.
- [3] Bostan, A., Jeannerod, C.-P., and Schost, É. (2008). Solving structured linear systems with large displacement rank. *Theoretical Computer Science*, 407(1):155–181.
- [4] Chronopoulos, A. T. (1991). s -step iterative methods for (non) symmetric (in) definite linear systems. *SIAM Journal on Numerical Analysis*, 28(6):1776–1789.
- [5] Csiszar, I. and Körner, J. (2011). *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Cambridge University Press.
- [6] Eisenstat, S. C., Elman, H. C., and Schultz, M. H. (1983). Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357.
- [7] Elman, H. C. (1982). *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations*. PhD thesis, Yale University.
- [8] Embree, M. (2003). The tortoise and the hare restart GMRES. *SIAM Review*, 45(2):259–266.
- [9] Golub, G. H. and Greif, C. (2003). On solving block-structured indefinite linear systems. *SIAM Journal on Scientific Computing*, 24(6):2076–2092.
- [10] Golub, G. H. and Van Loan, C. F. (2012). *Matrix Computations*. Johns Hopkins University Press, 3rd edition.

- [11] Greenbaum, A., Pták, V., and Strakoš, Z. (1996). Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17(3):465–469.
- [12] Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436.
- [13] Ho, K. L. and Greengard, L. (2012). A fast direct solver for structured linear systems by recursive skeletonization. *SIAM Journal on Scientific Computing*, 34(5):A2507–A2532.
- [14] Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282.
- [15] Liesen, J. and Tichý, P. (2004). Convergence analysis of Krylov subspace methods. *GAMM-Mitteilungen*, 27(2):153–173.
- [16] Paige, C. C. and Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629.
- [17] Pinsker, M. (1964). *Information and Information Stability of Random Variables and Processes*. Holden-Day, San Francisco.
- [18] Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition.
- [19] Saad, Y. and Schultz, M. H. (1985). Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Mathematics of Computation*, 44(170):417–424.
- [20] Saad, Y. and Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869.
- [21] Walker, S. G. (2016). An iterative algorithm for solving sparse linear equations. To appear in *Communications in Statistics-Simulation and Computation*.
- [22] Young, D. M. and Jea, K. C. (1980). Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra and Its Applications*, 34:159–194.